<table>
<thead>
<tr>
<th>Symbols</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td># Not equal to 87</td>
<td>87</td>
</tr>
<tr>
<td>$ String selector 68, 333</td>
<td>87</td>
</tr>
<tr>
<td>&amp; And 88</td>
<td>88</td>
</tr>
<tr>
<td>* Export mark 36</td>
<td>36</td>
</tr>
<tr>
<td>+ Multiplication 51, 54, 58</td>
<td>51, 54, 58</td>
</tr>
<tr>
<td>+ Addition 51, 54, 58</td>
<td>51, 54, 58</td>
</tr>
<tr>
<td>+ Concatenation 66</td>
<td>66</td>
</tr>
<tr>
<td>+ UML public 540, 563</td>
<td>540, 563</td>
</tr>
<tr>
<td>- Read-only export mark 82</td>
<td>82</td>
</tr>
<tr>
<td>- Subtraction 51, 54, 58</td>
<td>51, 54, 58</td>
</tr>
<tr>
<td>/ Division 51, 58</td>
<td>51, 58</td>
</tr>
<tr>
<td>:= Assignment 50</td>
<td>50</td>
</tr>
<tr>
<td>:= GCL assignment 69</td>
<td>69</td>
</tr>
<tr>
<td>:= Import abbreviation 544</td>
<td>544</td>
</tr>
<tr>
<td>&lt; Less than 87</td>
<td>87</td>
</tr>
<tr>
<td>&lt;= Less than or equal to 87</td>
<td>87</td>
</tr>
<tr>
<td>= Equal to 87</td>
<td>87</td>
</tr>
<tr>
<td>&gt; Greater than 87</td>
<td>87</td>
</tr>
<tr>
<td>&gt;= Greater than or equal to 87</td>
<td>87</td>
</tr>
<tr>
<td>^ Pointer dereference operator 468</td>
<td>468</td>
</tr>
<tr>
<td></td>
<td>Case separator 106</td>
</tr>
<tr>
<td>~ Not 89</td>
<td>89</td>
</tr>
<tr>
<td>– UML private 540, 563</td>
<td>540, 563</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Numerics</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X String sentinel 65, 333–335</td>
<td>333–335</td>
</tr>
<tr>
<td>100 Invariant violation 157</td>
<td>157</td>
</tr>
<tr>
<td>20 Precondition violation 120</td>
<td>120</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS 58</td>
<td>58</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>553</td>
</tr>
<tr>
<td>method attribute 553</td>
<td></td>
</tr>
<tr>
<td>objects and methods 578</td>
<td>578</td>
</tr>
<tr>
<td>record attribute 553</td>
<td>553</td>
</tr>
<tr>
<td>Abstract data structure 113, 129</td>
<td></td>
</tr>
<tr>
<td>list 130</td>
<td>130</td>
</tr>
<tr>
<td>stack 119</td>
<td>119</td>
</tr>
<tr>
<td>Abstract data type 113, 129</td>
<td></td>
</tr>
<tr>
<td>list 138</td>
<td>138</td>
</tr>
<tr>
<td>stack 124</td>
<td>124</td>
</tr>
<tr>
<td>Abstraction 6, 30, 113, 533–552</td>
<td></td>
</tr>
<tr>
<td>and nesting 151</td>
<td>151</td>
</tr>
<tr>
<td>behavior 551</td>
<td>551</td>
</tr>
<tr>
<td>class 538</td>
<td>538</td>
</tr>
<tr>
<td>computation 536</td>
<td>536</td>
</tr>
<tr>
<td>data 313, 533</td>
<td>313, 533</td>
</tr>
<tr>
<td>procedure 537</td>
<td>537</td>
</tr>
<tr>
<td>program 313</td>
<td>313</td>
</tr>
<tr>
<td>statement 536</td>
<td>536</td>
</tr>
<tr>
<td>structure 535</td>
<td>535</td>
</tr>
<tr>
<td>type 534</td>
<td>534</td>
</tr>
<tr>
<td>Actual parameter 37, 51</td>
<td></td>
</tr>
<tr>
<td>ADS. See Abstract data structure</td>
<td></td>
</tr>
<tr>
<td>ADT. See Abstract data type</td>
<td></td>
</tr>
<tr>
<td>Algol 2</td>
<td>2</td>
</tr>
<tr>
<td>Alphabet 15</td>
<td>15</td>
</tr>
<tr>
<td>closure of 16</td>
<td></td>
</tr>
<tr>
<td>nonterminal 17</td>
<td>17</td>
</tr>
<tr>
<td>terminal 17</td>
<td>17</td>
</tr>
<tr>
<td>Alternation, EBNF operation 25</td>
<td>25</td>
</tr>
<tr>
<td>Analysis</td>
<td></td>
</tr>
<tr>
<td>versus design 321</td>
<td>321</td>
</tr>
<tr>
<td>ANYREC 476</td>
<td>476</td>
</tr>
<tr>
<td>API. See Application programming interface</td>
<td></td>
</tr>
<tr>
<td>Application programming interface 4</td>
<td>4</td>
</tr>
<tr>
<td>ARRAY 65</td>
<td>65</td>
</tr>
<tr>
<td>one-dimensional 313–336</td>
<td></td>
</tr>
<tr>
<td>open 319</td>
<td>319</td>
</tr>
<tr>
<td>See also specific type, e.g.</td>
<td></td>
</tr>
<tr>
<td>CHAR, ARRAY OF</td>
<td></td>
</tr>
<tr>
<td>two-dimensional 381–392</td>
<td>381–392</td>
</tr>
<tr>
<td>ASSERT 144, 156, 157–163</td>
<td></td>
</tr>
<tr>
<td>purpose of 160</td>
<td>160</td>
</tr>
<tr>
<td>Assertion 156</td>
<td>156</td>
</tr>
<tr>
<td>See also ASSERT</td>
<td></td>
</tr>
<tr>
<td>Assignment</td>
<td>474, 554</td>
</tr>
<tr>
<td>class 474, 554</td>
<td>474, 554</td>
</tr>
<tr>
<td>formal definition of 146</td>
<td>146</td>
</tr>
<tr>
<td>in GCL 69</td>
<td>69</td>
</tr>
<tr>
<td>integer to real 53</td>
<td>53</td>
</tr>
<tr>
<td>multiple, in GCL 70</td>
<td>70</td>
</tr>
<tr>
<td>pointer 465, 470</td>
<td>465, 470</td>
</tr>
<tr>
<td>record 490</td>
<td>490</td>
</tr>
<tr>
<td>rule for extensible records 575</td>
<td>575</td>
</tr>
<tr>
<td>rule for parameter 557</td>
<td>557</td>
</tr>
<tr>
<td>statement 50</td>
<td>50</td>
</tr>
<tr>
<td>symbol 50</td>
<td>50</td>
</tr>
<tr>
<td>Associative operator 52</td>
<td></td>
</tr>
<tr>
<td>Attribute. See Unified Modeling Language</td>
<td></td>
</tr>
<tr>
<td>Automatic garbage collection 473</td>
<td>473</td>
</tr>
<tr>
<td>Average</td>
<td></td>
</tr>
<tr>
<td>computing from window 211</td>
<td>211</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backus, John 2</td>
<td>2</td>
</tr>
<tr>
<td>Binary search</td>
<td></td>
</tr>
<tr>
<td>iterative 357–360</td>
<td></td>
</tr>
<tr>
<td>recursive 433–435</td>
<td></td>
</tr>
<tr>
<td>Binary tree 515–528</td>
<td></td>
</tr>
<tr>
<td>abstract 515</td>
<td></td>
</tr>
<tr>
<td>depth of node 516</td>
<td>516</td>
</tr>
<tr>
<td>height 516</td>
<td>516</td>
</tr>
<tr>
<td>inorder traversal 518</td>
<td>518</td>
</tr>
<tr>
<td>insert operation 519</td>
<td>519</td>
</tr>
<tr>
<td>internal node 519</td>
<td>519</td>
</tr>
<tr>
<td>leaf 515</td>
<td>515</td>
</tr>
<tr>
<td>left child 515</td>
<td>515</td>
</tr>
<tr>
<td>postorder traversal 518</td>
<td>518</td>
</tr>
<tr>
<td>preorder traversal 517</td>
<td>517</td>
</tr>
<tr>
<td>right child 515</td>
<td>515</td>
</tr>
<tr>
<td>root 515</td>
<td>515</td>
</tr>
<tr>
<td>search tree 516, 581–592</td>
<td></td>
</tr>
<tr>
<td>Binomial coefficient 409</td>
<td>409</td>
</tr>
<tr>
<td>Bisection algorithm 214–217, 277–279</td>
<td></td>
</tr>
<tr>
<td>BlackBox framework 1–6</td>
<td>1–6</td>
</tr>
<tr>
<td>Bohm, Corrado 222</td>
<td>222</td>
</tr>
<tr>
<td>BOOLEAN 87</td>
<td>87</td>
</tr>
<tr>
<td>expression 87</td>
<td>87</td>
</tr>
<tr>
<td>variable 88, 96</td>
<td>88, 96</td>
</tr>
<tr>
<td>Bulletproof 144, 261</td>
<td></td>
</tr>
<tr>
<td>Button</td>
<td></td>
</tr>
</tbody>
</table>
command control 79
commander 40
dialog box 76
radio 103

C
Calling mode 126
by constant reference 126, 326–328
by reference 126, 263–265
by result 66, 101, 120, 126, 265–267
by value 126, 252–256, 261–262
default, See by value
for function procedure 287
summary of by value, by constant reference, by result, and
by reference 327
summary of by value, by result, and by reference 267
Caption control 79
CASE statement 106
CHAR 62
ARRAY OF 65, 333
number line 63
ordinal value 63
output of array to Log 66
output to Log 63
Check box 97
CHR 63
Class 175
assignment rule 474, 554
assignment rule for extensible records 575
assignment rule for parameters 557
Class composition 175, 191, 564–565
symbol for 182, 564
versus inheritance 573
Client module 30
Closure
of an alphabet 16
of the derivation operation 19
Code 101
Code folder 8
Commander button 40
Comments 34
Compilation unit 29
Compilers 31–32
Compiling a program 38
Component Pascal 2–3
calling mode 126
comments 34
identifier 34–36
program 32
reserved words 34
semicolon 38
statement 37
syntax 26–27, 603–604
Concatenation 16, 66
Constant 97
Context sensitive grammar 21
Control
caption 79
command button 79
dialog box 79
radio button 103
text field 79
Control guard 546
Controller 180
CP. See Component Pascal
Cross-platform 3–4

D
Dead code 161
DEC 56
Default calling mode. See Calling mode, by value
Delegation 584, 594
Derivation
of a valid sentence 19
Design
by contract 144, 577
by contract rule 370, 577
versus analysis 321
Dialog
.Par 548
.Update 83
.interface 548
Dialog box 75–84
button 76
control 79
from program 195
tool 353–355
I/O, See specific type e.g. String.
.output to dialog box
Directory 184
DIV 54, 58
do statement, GCL 227
Docu folder 8, 40, 81
Documentation files 40
Drag and drop 9
Dynamic storage allocation 463

E
EBNF. See Extended Backus-Naur form
ELSE 94
ELSIF 153, 164
Empty string 16
Enc folder 8
Encoding BlackBox documents 9–13
ENTIER 59, 258
eot technique 202
Error
.logical 39
.syntax 38–40
Evaluation
.full 350
.short-circuit 351
Execution count
.best-case versus worst-case 329, 351
.nested loop 240–242, 243–245
.single loop 204–208
Export
.read-only 82, 191, 563
.read/write 83, 563
Exporting procedures 36
Expression
.BOOLEAN 87
.infix 114
.INTEGER 54
.mixed 57
.postfix 114
.prefix 114
.REAL 51, 213
Extended Backus-Naur form 24–25
EXTENSIBLE 575
Index 607

F
Factorial
  iterative 281
  recursive 400
Factory design pattern 184
Field 82
  in trap window 124
Flowchart 94
  CASE 107
  IF with ELSE 96
  IF without ELSE 94, 95
  nested IF 153
  REPEAT 298
  sequential vs nested IF 157
  WHILE 201
FOR statement 222–227
Formal parameter 37, 51
Formal specification 145
  for array 337
Formatter 182
  called by constant reference 255
Full evaluation 350
Function procedure 60–62, 277–288

G
GCD. See greatest common divisor
GCL. See Guarded command language
Global data 29
Global variable 76, 269–272
  in trap window 124
  when to use 272
Grammar 17–22
  context sensitive 21
  for algebraic expressions 23
  for identifiers 18
  for signed integers 19
  four parts of 17
Graphical user interface 5
Greatest common divisor 407
Guard
  control 546
Guarded command language 68
  assignment statement 69
  do statement 227
  if statement 108, 165
  multiple assignment 70
  skip statement 108
  GUI. See Graphical user interface

H
HALT 161
has-a relationship 564
Heap 463
Hoare triple 145, 156, 338
Hollywood Principle 548

I
Identifier
  Component Pascal 34–36
  grammar for 18
Identity element 16
IF statement 92–103
  nested 151–165
if statement, GCL 108, 165
Import abbreviation 544
Importing procedures 36
IN. See Calling mode, by constant reference
INC 56
Infix expression 114
Inheritance 175, 474, 552–554
  symbol for 181, 564
  versus class composition 573
Input
  See specific type, e.g. REAL, input from dialog box
Insertion sort 438
INTEGER 53
  ARRAY OF 331
  expression 54
  input from dialog box 76
  output to dialog box 81
  output to Log 53
  range of 59
  to string conversion 66
Interactor 83
Interface 30–31
  See Module for interfaces of specific modules
Invariant 156–163
  implemented with ASSERT 160
  strong versus weak 159
Inverse operator 90
is-a relationship 564
Iterator 182

J
Jacopini, Guiseppe 222

L
Language 15–17
  definition of 17
Largest
  computing from array 328
  computing from window 212
LargestLast 330
Layout mode 79, 80, 83
Left-to-right rule 52
Lehmer generator 301
LEN 321, 325, 334
  with $ string selector 335
  with two-dimensional array 383
Length
  of ARRAY OF CHAR 334
  of ARRAY OF REAL 321, 325, 334
  of string 16
lg 360, 458
List 130
  abstract data structure 130
  abstract data type 138
  circular doubly-linked 488
  circular linked 477
  linked 469–508
  linked class 492, 592–601
Loading modules 76
Local data 29
Local variable 76
Logarithm 360, 458
Logical error 39
LONGINT
  range of 59
Loop invariant 208–209, 505

M
Mask mode 79
Math
  .Exp 60
  .IntPower 61, 214
  .Ln 60
Mutual recursion 422
MVC. See Model view controller

N
Name
of variable 49
Naur, Peter 2
Nesting 151, 239
NEW 465, 470
method attribute 553
NIL 464, 469
Nonterminal alphabet 17

O
Oberon 2
Oberon microsystems 3
URL 7
Object 175
Object program 31
Object-oriented language 6
Object-oriented programming 29
primary benefit of 562
shift in focus 540
terminology 179, 540
ODD 88
OOP. See Object-oriented programming
Open array 319
Operation. See Unified Modeling Language
Operator
associative 52
inverse 90
precedence 91
Optional, EBNF operation 25
OR 89
ORD 63
Ordinal value of CHAR 63
OUT. See Calling mode, by result
Output
See specific type, e.g. REAL, output to Log

P
Parameter
actual 37, 51
class assignment rule 557
formal 37, 51
Parsing 22
Pascal, Blaise 2
Pascal’s triangle 409
Pbox project 113
URL 64
PboxCLListADT
.Clear 478, 486
.Empty 478, 486
.GoNext 479, 486
.Insert 479, 486
.NodeContent 480, 486
implementation 485
interface 477
PboxListADS
.Clear 131
.Display 134
.GetLength 133
.InsertAtN 131
.Length 131
.RemoveN 132
.Search 133
.T 131
interface 130
PboxListADT
.InsertAtN 139
.List 139
.RemoveN 145
implementation 375
interface 138, 375
PboxListObj
documentation 493–94
implementation 498–499
interface 492, 593
PboxListSta
.EmptyNode.DisplayN 598
.EmptyNode.InsertAtN 600
.List 598
.List.InsertAtN 598
.List.Search 600
.List.SearchN 600
.NonEmptyNode.DisplayN 598
.NonEmptyNode.InsertAtN 599
implementation 595
interface 592
UML class diagram 594
PboxMappers
Index 609

(formatter, write real, write real matrix, write real vector, write string, connect to, connect to 187, 192, 193, 210, 192, 193, 210, 202, 210, 382, 318, 210, 202, 210, 193, 202, 210, 382, 318) interface 182–183

PboxPizza
alternate design 572
implementation 569
UML class diagram 568

PboxRandom
 interv 298
randomize 294
real 296
set seed 294
implementation 302
interface 293

PboxShapeAbs
circle 554
circle.set id string 561
rectangle 553
rectangle.set id string 554, 561
rectangle.set state 554
shape 553
shape.set id string 553
implementation 559
interface 552

PboxShapeObj
implementation 548
interface 541

PboxStackADS
capacity 119
clear 119
num items 120
pop 120, 371
push 120, 369
implementation 370
interface 119, 368

PboxStackADT
clear 126
pop 126
stack 125

PboxStackObj
implementation 372
interface 176, 371

PboxStrings
int to string 66
lower 64
real to string 84
to lower 101
upper 64
interface 64
url 64

PboxTreeADT
insert 520
t 520
tree 520

PboxTreeObj
implementation 525
interface 521, 582

PboxTreeSta
empty node.insert 590
empty node.pre order 588
node.insert 589
non empty node.insert 589
non empty node.pre order 589
tree.clear 587
tree.insert 589
tree.pre order 588
implementation 585
interface 582
UML class diagram 583

Performance. See Execution count
Permutation 340, 417
Plato 533
POINTER 463
assignment 465, 470
legal operations 468
period abbreviation for ^, 468
versus record 574

Polymorphic dispatch 558
Polymorphism 475, 555–558, 585
pop operation 114
post condition 120, 145
postfix expression 114
precedence of operators 91, 116
pre condition 120, 145, 156
prefix expression 114
Primitive type 125
Private 563
versus public 576
Procedure
function 60–62, 277–288
proper 42–45, 251–272
signature 37

Production
rules of 17, 18
Program 32
compiling 38
style 41
Programmer-defined type 125
Project folders 6–8
Proper procedure 42–45, 251–272
Public 563
versus private 576
Push operation 114
Q
Queue 143
Quick sort
concept 437
correctness of 446
implementation 439–446
R
Radio button 103
Random
integers 296
number generators 300
reals 294
Range
of INTEGER 59
of LONGINT 59
of REAL 60
Read-only export 82, 191, 563
Read/write export 83, 563
REAL 49
伎 ARRAY OF 314
expression 51, 213
input from dialog box 84
input from window 190
input of matrix from window 382
input of vector from window 318
output of matrix to window 383
output of vector to window 326
output to Log 50, 52
output to window 189
range of 60
Receiver 176
restrictions on 500
RECORD 82
assignment 490
versus pointer 574
Recursion 399–423
cost of 423
macroscopic viewpoint 405
microscopic viewpoint 404
mutual 422
REPEAT statement 298
Repetition, EBNF operation 25
Republic 533
Reserved words 34
Return address 251, 280, 284
RETURN statement 278
in proper procedure 358
Rigid variable 146, 338
Rsrc folder 8, 80
Rules of production 17, 18
Run-time stack
for function procedures 280, 401
for proper procedures 251
S
Scanner 182
Schrage’s algorithm 305
Searching
iterative 349–360
recursive 433–435
Selection sort 360–362
concept 438
Selector, $ 68, 333
Semantics 15
Semicolon
rules for 96
Semicolon seperator 38
Sequential search 350–356
Server module 30
SHORT 59, 257
Short-circuit evaluation 351
Signature of procedure 37
skip statement, GCL 108
Socrates 533
Sorting
complexity 457
iterative 360–362
recursive 435–458
Source program 31
Specification, formal 145
for array 337
Stack 114
abstract data structure 119
abstract data type 124
class 176
See also Run-time stack
Stack frame 255
Start symbol 17, 18
State design pattern 581–601
Statement
See specific statement, e.g. IF statement
StdCmds
.OpenAuxDialog 81
.OpenToolDialog 354
StdLog
.Char 63
.Real 51
.String 37, 66
interface 31
Stepwise refinement 217–222, 245–247
String
alphabetic order 99
empty 16
length of 16
output to dialog box 84
output to Log 37
output to window 186
selection with 99
selector $ 68, 333
Structured programming theorem 222
Stub 497
Style
precondition 120
program 41
Subclass 474, 553
Superclass 474, 553
Sym folder 8
Syntax 15
Component Pascal 26–27, 603–604
error 38–40
T
Terminal alphabet 17
Text field control 79
Text subsystem 8–9
TextControllers 181
.Controller 191
.Controller.text 192
.Focus 191
interface 190–191
TextModels 181
.dir 185
dir.New 187
.Directory 184
.Directory.New 184
.Model 184
interface 184
TextViews 181
dir.New 187
.Directory.New 185
interface 185
Tool dialog box 353–355
Towers of Hanoi 419
Trap 122, 260, 317
Type
base 552
extension 552
guard 475
in GCL 69
of variable 49
primitive 125
programmer-defined 125
static versus dynamic 475, 558
See also specific type, e.g. REAL
U
UML. See Unified modeling language
Unified Modeling Language 181, 474, 562–565
attribute 540
class diagram 539, 563
operation 540
URL
Oberon microsystems 7
PboxStrings 64

V
Value
  of variable 49
VAR. See Calling mode, by reference
Variable
  global 76
  local 76
  name of 49
  rigid 146, 338
  three attributes of 49
  type of 49
  value of 49
  See also specific type, e.g. REAL
View 179
Views 181
  .OpenView 186, 187
  .View 186
  interface 186

W
WHILE statement 201–222
Window 179–197
  I/O, See specific type e.g. String,
    output to window
Wirth, Niklaus 2